Chapter One: Contents

(Shared Rides and Long Walks to School – 10 December 2002 – LA-UR 01-5713 – Portland Study Reports)

Chapter One—Shared Rides and Long Walks to School

1. Fixing Shared Ride Problems from Incomplete Survey Match

The TRANSIMS Activity Generator (see the general documentation) determines the initial household activity patterns by matching a synthetic household with a survey household based on household and person demographics. Each individual in the survey household does not necessarily match every individual in the synthetic household. This can produce problems assigning intra-household shared rides since the driver's activity pattern from the survey household may not be present in the synthetic household.

Example:

If the matching criteria is number of workers and household size, the following households could be matched using the above criteria:

- Synthetic HH - 4 person: 1 worker, 1 adult and 3 children.
- Survey HH - 4 person: 1 worker, 2 adults and 2 children.

The worker from the synthetic household would receive the activity pattern from the worker in the survey household. The children in the synthetic household receive the activity patterns of the children in the survey household with one child's activity pattern repeated for the third child in the synthetic household. The activity pattern for the second adult in the survey household is not used in the synthetic household.

This unmatched survey adult is the driver of the shared rides that transport the survey children to activities throughout the day. The children in the synthetic household are passengers that need to match with a driver's activity pattern that does not exist in the synthetic household.

The Activity Generator detects this condition and reports a problem (type 1) in the problem file, then changes the mode of the children's activities to an inter-household shared ride, which we call "magic move".

This problem can be corrected in most of the synthetic households by iteratively selecting another survey household match. The Activity Regenerator has a command (R) that requests another survey household match to obtain a different activity pattern for the synthetic household. The Activity Regenerator reports a problem of type 1 in the problem file for any remaining households that were not corrected by the survey household rematch. This problem file is used as the basis for the next iteration where the rematch is again requested for the remaining problem households.

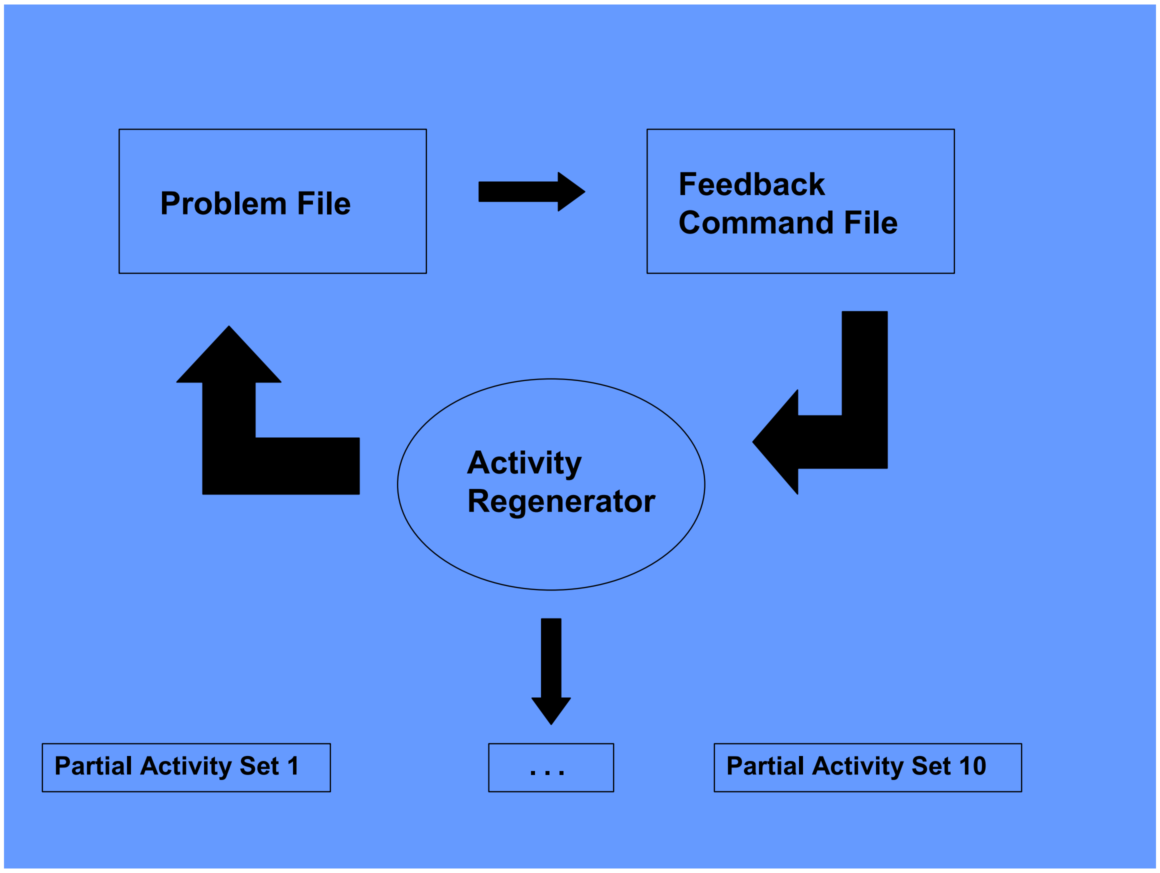

Fig. 1. Process to fix shared ride problems caused by incomplete survey household match.

Fig. 1 illustrates the steps in the process:

- Use the Activity Generator problem file to determine which households have problem type 1 (Bad Shared Ride Match).
- Create a feedback command file for the Activity Generator that has a rematch request for each of these households.
- Run the Activity Regenerator to resample the survey households, generate a new activity pattern, and report remaining households with the problem in the Activity Generator problem file.

Repeat the process until few households remain with this problem.

This process, along with the school bus corrections (see Section 4 of this Chapter), was applied to activity set AS-1 (see Fig. 3 in Volume One (*Introduction/Overview*), Chapter Three (*The Modules*)) to produce activity set AS-2. After the initial Portland activity generation creating AS-1, 11,942 households had this problem. After 10 iterations were completed, the problem was reported in 294 households.

Table 1. Number of households with shared ride problems by iteration.

Iteration	Number Problem Households
0	11941
1	3752
2	2012
3	1327
4	972
5	735
6	603
7	476
8	406
9	346
10	294

2. Implementation

2.1 Identifying the Problem Households

The first step in the process is to extract the problem households from the Activity Generator problem file. Multiple problem files may exist from the initial activity generation if the Activity Generator was executed in parallel. These files should be concatenated to form a single problem file that will be the basis for the shared ride iterations. The problem as reported in the problem file has the following format:

```
1 3 <hh> <person> <activity>
```

Example:

```
1 3 4389 8476 12
```

The Unix utility *awk* is used to parse the problem file and create a feedback command file that requests a survey household rematch for the problem households in the file. A household may have multiple entries in the problem file; however, only one rematch request is needed per household. The Unix utility *sort* with the command line argument to remove duplicate lines is used to remove the duplicate household entries.

The format of the Activity Regenerator rematch command is

```
<household id> R
```

The following command sequence is used to create the feedback command file from the problem file.

```
% /usr/bin/awk '{if ($1==1) print $3 " R"}' $PROBLEM_FILE | /bin/sort -n -u > $FEEDBACK_COMMANDS
```

where `PROBLEM_FILE` is the starting problem file produced by the Activity Generator and `FEEDBACK_COMMANDS` is the name of the feedback command file that is produced.

2.2 Iteratively Regenerating Activities From a Survey Household Rematch

The Activity Regenerator uses configuration file keys to specify the name of the feedback command file (`ACT_FEEDBACK_FILE`), the name of the problem file that is produced (`ACT_PROBLEM_FILE`), and the name of the file containing the regenerated activities (`ACT_PARTIAL_OUTPUT`). If a key is specified multiple times in the configuration file, the last specification overrides all other specifications of the key. This feature is used to iteratively change the name of the feedback file, the problem file, and the regenerated activities file by appending the appropriate key values for the iteration to the configuration file, then starting the Activity Regenerator for the next iteration.

Ten iterations were used for the Portland activity set. The following illustrates the main loop through the iterations:

- Append the appropriate configuration file key values to the configuration file.
- Run the Activity Regenerator.
- Extract the remaining problem households from the problem file produced by the Activity Regenerator and produce a feedback command file for the next iteration
- Increment the iteration counter and go to next iteration.

```
# Do 10 iterations
for each i (1 2 3 4 5 6 7 8 9 10)
   echo "Activity Regenerator for shared rides iteration " $i >>&!
$LOG_DIR/ActivityRegenerator.log
   echo "ACT_FEEDBACK_FILE " $FEEDBACK_COMMANDS.$ITER >> $TMP_CONFIG
   echo "ACT_PARTIAL_OUTPUT " $PARTIAL_FILE.$i >> $TMP_CONFIG
   echo "ACT_PROBLEM_FILE " $PROBLEM_FILE.$i >> $TMP_CONFIG
   $TRANSIMS_HOME/bin/ActivityRegenerator $TMP_CONFIG >>&!
$LOG_DIR/ActivityRegenerator.log
   setenv ITER `echo $ITER+1 | /usr/bin/bc`
                           /usr/bin/awk '{if ($1==1) print $3 " R"}'
$PROBLEM_FILE.$i | /bin/sort -n -u > $FEEDBACK_COMMANDS.$ITER
end
```

3. Creating a Merged Activity Set

After the iterations are complete, ten files with regenerated activities for problem households are produced. These files must be combined with the original activity set to produce a "corrected" activity set. A single household may appear multiple times in the regenerator activity sets. The combined activity set should contain the most recent activity set for a regenerated household. The regenerated activities are combined with the original activities by creating a merged index and then using the merged index to create a merged activity file. The Activity Regenerator creates a household index for each of the regenerated activity files. The *MergeIndices* program uses these index files and the household index for the original activity set to produce a merged index. The index files to merge are command line arguments to the program. The order of the arguments is important. *MergeIndices* combines the indexes in order of their appearance on the command line. The entries for a household found later on the command line will override the entries for the same household in an index file processed previously.

```
$TRANSIMS_HOME/bin/MergeIndices $ACTIVITY_FILE.merged.hh.idx
$ACTIVITY_FILE.hh.idx
$PARTIAL_FILE.1.hh.idx
$PARTIAL_FILE.2.hh.idx
$PARTIAL_FILE.3.hh.idx
$PARTIAL_FILE.4.hh.idx
$PARTIAL_FILE.5.hh.idx
$PARTIAL_FILE.6.hh.idx
$PARTIAL_FILE.7.hh.idx
$PARTIAL_FILE.8.hh.idx
$PARTIAL_FILE.9.hh.idx
$PARTIAL_FILE.10.hh.idx
```

where *$ACTIVITY_FILE.merged.hh.idx* is the name of the merged index that will be produced, *$ACTIVITY_FILE.hh.idx* is the household index of the original activity set, and *$PARTIAL_FILE.?.hh.idx* are the household indexes of the regenerated activity sets named in the order in which they were produced.

A merged activity file is produced from the merged index using the *IndexDefrag* program.

```
$TRANSIMS_HOME/bin/IndexDefrag $ACTIVITY_FILE.merged.hh.idx
$ACTIVITY_FILE.sfb
```

where *$ACTIVITY_FILE.merged.hh.idx* is the name of the merged index file created by *MergeIndices*, and *$ACTIVITY_FILE.sfb* is the name of the merged activity set that will be produced.

Refer to the *FixSharedRides* script in Volume Seven (*Appendix: Scripts, Configuration Files, Special Travel Time Functions*).

4. Fixing Long Walk/Bike Trips to School

The initial activity set from the Activity Generator will have some persons who have long walks or bicycle trips to school. This occurs because the synthetic households have activity and initial mode patterns from the survey households. Although the survey household locations may have been within walk/bike distance of school, the synthetic household has different locations that make the walk/bike mode choice infeasible. The solution is to identify these trips and change the mode to school bus.

4.1 Creating an Iteration Database

Create an iteration database from the activity set that contains the appropriate fields to identify the long walk/bike trips. The Collator is used to create the database. Trips in the iteration database consist of the travel that occurred between the starting activity location and the end activity location. The information needed is the starting activity location, the end activity location, the type of the end activity, the mode used to arrive at the end activity, and the distance between the starting and ending activity locations.

4.2 Identifying Long Walk/Bike Trips to School

The ending activity type field in the iteration database identifies school trips (type = 7), and the Euclidean distance field gives the distance in meters between the starting location of the trip and the school (end) location. The mode field indicates if the trip was walk (mode = 1) or bike (mode = 7).

For those walk trips greater than 2,500 meters and for bicycle trips greater than 7,500 meters, format an Activity Regenerator command to change the mode to school bus (mode 9). Store the commands in a feedback command file that will be used by the Activity Regenerator.

4.3 Mode Change to School Bus using Activity Regenerator

The Activity Regenerator uses the feedback command file and changes the mode of the travelers with long walk/bikes to school to school bus. The partial activity file from the Regenerator will contain the households with mode changes and must be merged back into the original activity set.

The initial activity set contained 9,819 long walk/bike trips to school. The mode of these trips was changed to school bus (mode 9).

5. IMPLEMENTATION

5.1 Creating an Iteration Database

The Collator is run in parallel to create iteration databases that are then combined to form a single iteration database. The number of Collator processes and hence, the number iteration databases produced, may be sized to the number of processors available. Ten Collator processes were used to create the iteration databases. The *MakeHouseholdFile* program was used to split the households in the population (and the activity set) into 10 portions. The household files are named with a special suffix *(.tXX*) where *XX* is a conversion of a number into an alphabetic string. The numbering starts at 0, which converts to AA, then sequentially increases (1 = AB, 2 = AC, ...). Household files *hh.tAA* - *hh.tAJ* are produced by the *MakeHouseholdFile*. The Collator is started on 10 processors with a numerical rank from 0 to 9 as a command line argument. The Collator processes will look for the appropriate household file and create an iteration database containing information for those households only. The separate databases are concatenated into a single database with the header lines removed from the second to nth database.

Use the following configuration file keys in the configuration file for the Collator to produce the appropriate data columns:

```
SEL_USE_START_ACT_TYPE,
SEL_USE_START_ACT_LOCATION,
SEL_USE_START_MODE_PREF,
SEL_USE_END_MODE_PREF,
SEL_USE_END_ACT_TYPE,
SEL_USE_END_ACT_LOCATION,
SEL_USE_EUCLID.
```

Refer to the script *RunCollatorInParallel* in Volume Seven (*Appendix*)

5.2 Identifying Long Walk/Bike Trips to School

A perl script is used to parse the iteration database and identify the long walk/bike trips to school. For each identified trip, an Activity Regenerator command is created and written to a file. The Regenerator command used is MS, which changes the mode for a single activity. Format of the command that appears in the feedback command file is

```
<hhid> <activity id> MS 9
```

Refer to the *FixSchoolWalk* and *FindLongWalks.pl* scripts in Volume Seven (*Appendix: Scripts, Configuration Files, Special Travel Time Functions*).

5.3 Mode Change to School Bus using Activity Regenerator

Run the Activity Generator using the mode change feedback commands generated above. The Activity Regenerator creates a partial activity set that contains only those households that have the requested mode changes. This partial activity set is then merged with the starting activity set to create an activity set that contains the changed households along with the other unchanged households in the original activity set. The *MergeIndices* and *IndexDefrag* programs are used to do the merging.